Introduction to numerical analysis and numerical methods for ordinary differential equations

LIM, Roktaek
Ulsan National Institute of Science and Technology

2025 Winter School on Numerical Relativity and Gravitational Waves

Part 1. Numerical analysis

What is numerical analysis?

The study of **algorithms** for the problems of continuous mathematics.

(Lloyd N. Trefethen, *SIAM News*, 1992)

What is numerical analysis?

The study of **algorithms** for the problems of continuous mathematics.

(Lloyd N. Trefethen, *SIAM News*, 1992)

The field concerned with the design of **computable algorithms** for solving mathematical problems, and with the analysis of their accuracy, efficiency, and other aspects of performance.

(D. Arnold, 2024 Simons Conference on Localization of Waves)

Accuracy

In the analysis of an algorithm, we should consider the error in computed approximations to the solution of our problem. This error may be due to various factors including rounding of arithmetic and the termination of an infinite process.

Accuracy

In the analysis of an algorithm, we should consider the error in computed approximations to the solution of our problem. This error may be due to various factors including rounding of arithmetic and the termination of an infinite process.

The composite midpoint rule

Let a and b be two real numbers with $a < b$. A continuous function $f : [a, b] \to \mathbb{R}$ is given.

$$\int_a^b f(x)\, dx \approx Q_{\mathsf{mid}}(f) = \sum_{j=1}^{N} h f\left(\bar{x}_j\right)$$

where $h = (b - a)/N$ and $\bar{x}_j = a + (j - 0.5)h$ for $j = 1, \ldots, N$.

$$\left| \int_a^b f(x)\, dx - Q_{\mathsf{mid}}(f) \right| \leq M \frac{(b-a)}{24} h^2.$$

for some constant M.

Efficiency

We usually measure the efficiency of a numerical algorithm in terms of the number of basic arithmetical operations required for a given accuracy in the result.

Efficiency

We usually measure the efficiency of a numerical algorithm in terms of the number of basic arithmetical operations required for a given accuracy in the result.

Complexity of matrix-vector multiplication

Matrices with rank 1 can always be obtained as the outer product of two vectors. Given $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^n$, the matrix with rank 1 can be computed by $A = \mathbf{a}\mathbf{b}^T$. Consider $A\mathbf{x}$ for some $\mathbf{x} \in \mathbb{R}^n$. This can be computed in two ways:

- $\mathbf{y} = (\mathbf{a}\mathbf{b}^T)\mathbf{x}$, the computational cost is $\mathcal{O}(mn)$,
- $\mathbf{y} = \mathbf{a}(\mathbf{b}^T\mathbf{x})$, the computational cost is $\mathcal{O}(m) + \mathcal{O}(n)$.

Stability

An algorithm can be viewed as another mapping $\hat{F} : X \to Y$. If $\hat{F}$ for a given F is accurate for each $x \in X$,

$$\frac{|F(x) - \hat{F}(x)|}{|F(x)|} \leq \epsilon.$$

Stability

An algorithm $\hat{F}$ for a problem F is stable if for each $x \in X$,

$$\frac{|F(\hat{x}) - \hat{F}(x)|}{|F(\hat{x})|} \leq \epsilon$$

for some $\hat{x}$ with

$$\frac{|\hat{x} - x|}{|x|} \leq \epsilon.$$

Stability

- Input data can be perturbed. However, they are in a neighborhood of exact input x, $|\hat{x} - x|/|x| \leq \epsilon$.
- Thus, any such $\hat{x}$ has to be considered as virtually equal to x.
- An algorithm is said to be **stable** if small errors in the inputs and at each step lead to small errors in the solution.
- If an algorithm is stable, $\hat{F}(x)$ is in a neighborhood of $F(\hat{x})$.
- An algorithm that amplifies errors is called **unstable**.

Errors

Absolute and relative errors

Let $\hat{x}$ be an approximation to a real number x. Then its absolute error is given by

$$E_{\text{abs}}\left(\hat{x}\right) = \left|x - \hat{x}\right|$$

and its relative error is defined as

$$E_{\text{rel}}\left(\hat{x}\right) = \frac{\left|x - \hat{x}\right|}{\left|x\right|}.$$

Sources of error: Rounding

Computers can only store finitely many quantities. Thus, finitely many real and complex numbers are representable on computers.

Let $F = \{x_1, \ldots, x_N\}$ be the ordered set of all representable numbers on computers. Suppose that a real number x does not have an exact representation and $x_{j-1} < x < x_j$. The process of replacing the real number x by a nearby machine number (either x_{j-1} or x_j) is called **rounding**, and the error involved is called **roundoff error**.

Floating point number system

A floating point number system $F \subset \mathbb{R}$ is a subset of the real numbers whose elements have the form

$$x = (-1)^s \times b^{(q-p)} \times m.$$

This notation has the following parts:

- s is 0 or 1,
- b is the base,
- q is any integer $e_{\min} \leq (q-p) \leq e_{\max}$, p is the precision, and
- m is a number represented by a digit string of the form within $[1, b)$.

Rounding

If $x \in \mathbb{R}$ then $\mathrm{fl}(x)$ denotes an element of F nearest x, then **rounding** is the mapping $x \to \mathrm{fl}(x)$.

Machine epsilon

The machine epsilon is defined as the difference between 1.0 and the smallest representable number which is greater than one. Let ϵ_{M} denote the machine epsilon.

The largest relative error

If $x \in \mathbb{R}$ lies in the range of F then

$$\mathrm{fl}(x) = x(1+\delta),\ |\delta| < \frac{1}{2}\epsilon_{\mathrm{M}}.$$

$$\left|\frac{\mathrm{fl}(x) - x}{x}\right| \leq \frac{1}{2}\epsilon_{\mathrm{M}}.$$

Model of arithmetic

To carry out rounding error analysis of an algorithm we need to make some assumptions about the accuracy of the basic arithmetic operations.

Standard model

There is a small positive number such that for the elementary arithmetic operations holds

$$\mathrm{fl}(x \star y) = (x \star y)(1+\delta),\ |\delta| \leq \frac{1}{2}\epsilon_{\mathrm{M}},\ \star = +, -, \times, /.$$

Relative roundoff errors of elementary operations bounded by $\frac{1}{2}\epsilon_{\mathrm{M}}$.

Sources of error: Truncation

Truncation errors result from the use of an approximation in place of an exact mathematical procedure.

Sources of error: Truncation

Truncation errors result from the use of an approximation in place of an exact mathematical procedure.

Forward difference approximation

We can compute the derivative of a function $f(x)$ at a point x_0 by using the forward difference,

$$D(f,h) = \frac{f(x_0+h) - f(x_0)}{h} \approx f'(x_0).$$

The approximation can be derived by using a Taylor series. Expand $f(x_0+h)$ around x_0:

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2} f''(x_0) + \mathcal{O}(h^3).$$

Then,

$$f'(x_0) = \frac{f(x_0+h) - f(x_0)}{h} - \frac{h}{2} f''(x_0) + \mathcal{O}(h^2).$$

Computational error and data error

Consider the evaluation of a function $f : \mathbb{R} \to \mathbb{R}$ for a given argument x:

- x: true value of input,
- $f(x)$: corresponding output value for true function,
- $\hat{x}$: approximate input actually used,
- $\hat{f}$: approximate function actually used.

Total error is given by

$$\hat{f}(\hat{x}) - f(x) = \hat{f}(\hat{x}) - f(\hat{x}) + f(\hat{x}) - f(x).$$

$\hat{f}(\hat{x}) - f(\hat{x})$ is the computational error and $f(\hat{x}) - f(x)$ is the propagated data error.

Roundoff error and truncation error

- Truncation error: difference between true result for true value of input and result produced by the given algorithm using exact arithmetic.
- Roundoff error: difference between result produced by the given algorithm using exact arithmetic and produced by the same algorithm using limited precision arithmetic.
- Computational error is sum of truncation error and roundoff error.

Part 2. ODEs

Initial value problems (IVPs)

An initial value problem (IVP) for a first-order ordinary differential equation is given by

$$y'(t) = f(t, y), \; y(t_0) = y_0,$$

$y'(t) = f(t, y)$ is the ordinary differential equation for $y(t)$ and $y(t_0) = y_0$ is the initial condition.

- $t_0, t_1, \ldots, t_N$: the points where the approximate solutions are defined.
- $y(t_j)$: the exact solution to the problem at t_j.
- y_j: the approximate solution at t_j.
- h_j: the step size, $h_j = t_{j+1} - t_j$; if we use the fixed step size, $h = \frac{t_N - t_0}{N}$

The goal is to compute an approximate solution $\{y_0, y_1, \ldots, y_N\}$ such that

$$y_j \approx y(t_j).$$

One-step method

The approximate solution is computed by

$$y_{j+1} = y_j + h_j \Phi\left(t_j, y_j, h_j\right) \text{ for } j = 0, \dots, N-1.$$

Explicit Euler method

In the Explicit Euler method,

$$\Phi\left(t_j, y_j, h_j\right) = f(t_j, y_j).$$

This is derived from the Taylor series expansion of $y(t)$ around t_j,

$$\begin{aligned} y(t_{j+1}) &= y(t_j) + h_j y'(t_j) + \frac{h_j^2}{2} y''(\xi_j) \\ &= y(t_j) + h_j f(t_j, y(t_j)) + \frac{h_j^2}{2} y''(\xi_j) \approx y(t_j) + h_j f(t_j, y(t_j)) \end{aligned}$$

where $\xi_j \in [t_j, t_{j+1}]$.

Local truncation error

The local truncation error is the difference between the approximate solution y_{j+1} and the solution at t_{j+1} of the ODE. The local truncation error for the Explicit Euler method is

$$\begin{aligned} e_{j+1} &= y(t_{j+1}) - y_{j+1} \\ &= y(t_{j+1}) - [y(t_j) + h_j f(t_j, y(t_j))] = \frac{h_j^2}{2} y''(\xi_j). \end{aligned}$$

The local truncation error is the left-over when plugging the exact solution into the approximate formula. The local truncation error e_{j+1} is the error done in one step when the approximation starts at the exact solution $y(t_j)$.

Global error

The global error is the difference between the exact $y(t_j)$ and the approximate solution y_j at t_j. The global error of the the approximate solution $\{y_0, y_1, \ldots, y_N\}$ in $[t_0, t_N]$ is

$$E_N = \max_{j=0,\ldots,N} |y(t_j) - y_j| \,.$$

If $e_j(h) = \mathcal{O}(h^{p+1})$, then $E_j(h) = \mathcal{O}(h^p)$.

Some examples of one-step methods

We consider the following IVP

$$y'(t) = f(t, y(t)),\ y(0) = y_0.$$

Then

$$y(t_{j+1}) = y_j + \int_{t_j}^{t_{j+1}} f(\tau, y(\tau))\, d\tau.$$

We can approximate the integral by means of $N+1$-point quadrature formula with nodes $c_0, \ldots, c_N$ and weights $w_0, \ldots, w_N$. The $N+1$-point quadrature formula can be written as follows:

$$\int_a^b f(x)\, dx \approx Q(f) = \sum_{j=0}^{n} w_j f(c_j).$$

Some examples of one-step methods

Trapezoidal rule is given by

$$Q_{\mathsf{trap}}(f) = \frac{b-a}{2}\left(f(a) + f(b)\right).$$

Since $y(t_{j+1})$ is approximated by the explicit Euler method, we can obtain

$$k_1 = f(t_j, y_j),\ k_2 = f(t_j + h, y_j + hk_1),\ y_{j+1} = y_j + \frac{h}{2}(k_1 + k_2).$$

We can also have the implicit Trapezoidal method, given by

$$y_{j+1} = y_j + \frac{h}{2}(f(t_j, y_j) + f(t_{j+1}, y_{j+1})).$$

Some examples of one-step methods

Midpoint rule is given by

$$Q_{\mathsf{mid}}(f) = (b-a) f\left(\frac{a+b}{2}\right).$$

Since $y(t_{j+1})$ is approximated by the explicit Euler method, we can obtain

$$k_1 = f(t_j, y_j),\ k_2 = f\left(t_0 + \frac{h}{2}, y_0 + \frac{h}{2} k_1\right),\ y_{j+1} = y_j + hk_2.$$

We can also have the implicit midpoint method,

$$y_{j+1} = y_j + hf\left(t_0 + \frac{h}{2}, \frac{y_j + y_{j+1}}{2}\right).$$

Some examples of one-step methods

Simpson's quadrature rule is given by

$$Q_{\mathsf{S}}(f) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right).$$

Then, we can obtain the fourth-order Runge Kutta method (RK4) as follows:

$$\begin{aligned}
k_1 &= f\left(t_j, y_j\right), \\
k_2 &= f\left(t_j + \frac{h}{2}, y_j + \frac{h}{2}k_1\right), \\
k_3 &= f\left(t_j + \frac{h}{2}, y_j + \frac{h}{2}k_2\right), \\
k_4 &= f\left(t_j + \frac{h}{2}, y_j + hk_3\right), \\
y_{j+1} &= y_j + \frac{h}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right).
\end{aligned}$$

Model problem

Consider the following IVP

$$y'(t) = \lambda y(t), \; y(0) = 1.$$

The exact solution to the problem is $y(t) = e^{\lambda t}$. The explicit Euler method with a fixed step size h gives

$$y_{j+1} = y_j + \lambda h y_j = (1 + \lambda h) y_j.$$

Then,

$$y_j = (1 + \lambda h)^j y_0 = (1 + \lambda h)^j.$$

In particular, $|y_j| \to 0$ if

$$h < \frac{2}{|\lambda|}.$$

If $\lambda < 0$, the condition $h < \frac{2}{|\lambda|}$ must hold. If $\lambda < 0$ and $|\lambda| \gg 1$, we should use very small step size h.

Stiff ODEs

The stiffness is not precisely defined. There are some statements describing a stiff problem.

- A problem is stiff if it contains widely varying time scales, i.e., some components of the solution decay much more rapidly than others.
- A problem is stiff if the stepsize is dictated by stability requirements rather than by accuracy requirements.
- A problem is stiff if explicit methods don't work, or work only extremely slowly.

The implicit Euler method for stiff ODEs

For the implicit Euler method

$$y_{j+1} = y_j + hf(t_{j+1}, y_{j+1}) = y_j + \lambda h y_{j+1}$$

we can obtain

$$y_{j+1} = \frac{1}{(1 - \lambda h)}.$$

Thus, $|y_j| \to 0$ when $|1 - \lambda h| > 1$. If $\lambda < 0$, then $|y_j| \to 0$ for any positive value of h.

Part 3. Exercises

Derive the error bound for the composite midpoint rule

Let a and b be two real numbers with $a < b$. A continuous function $f : [a, b] \to \mathbb{R}$ is given.

$$\int_a^b f(x)\, dx \approx Q_{\text{mid}}(f) = \sum_{j=1}^{N} h f\left(\bar{x}_j\right)$$

where $h = (b-a)/N$ and $\bar{x}_j = a + (j - 0.5)h$ for $j = 1, \ldots, N$.

$$\left| \int_a^b f(x)\, dx - Q_{\text{mid}}(f) \right| \leq M \frac{(b-a)}{24} h^2.$$

for some constant M.

Computational cost of matrix-vector multiplication

Matrices with rank 1 can always be obtained as the outer product of two vectors. Given $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^n$, the matrix with rank 1 can be computed by $A = \mathbf{a}\mathbf{b}^T$. We try to compute $A\mathbf{x}$ for some $\mathbf{x} \in \mathbb{R}^n$. Compute the computational costs of the following methods:

- $\mathbf{y} = (\mathbf{a}\mathbf{b}^T)\mathbf{x}$,
- $\mathbf{y} = \mathbf{a}(\mathbf{b}^T\mathbf{x})$.

Machine epsilon

$F \subset \mathbb{R}$ is a subset of real numbers whose elements have the form

$$x = (-1)^s \times b^{(q-p)} \times m.$$

If $x \in \mathbb{R}$ lies in the range of F then

$$\mathrm{fl}(x) = x(1+\delta),\ |\delta| < \frac{1}{2} b^{(1-p)}.$$

Roundoff error in the difference formula

We can compute the derivative of a function $f(x)$ at a point x_0 by using the forward difference,

$$D(f,h) = \frac{f(x_0+h) - f(x_0)}{h} \approx f'(x_0).$$

For any evaluation of the function,

$$\hat{f} = f(1+\delta),\ |\delta| < \frac{1}{2}\epsilon_{\mathrm{M}}.$$

Write $\hat{D}(f,h)$ and compute the error bound of $|D(f,h) - \hat{D}(f,h)|$.

Local truncation error

Compute the local truncation errors for the following methods:

- Implicit Euler method:

$$y_{j+1} = y_j + h_j f(t_{j+1}, y_{j+1}).$$

- Trapezoidal method

$$y_{j+1} = y_j + \frac{h_j}{2} \left(f(t_j, y_j) + f(t_{j+1}, y_{j+1}) \right).$$

Trapezoidal method for the stiff ODEs

The trapezoidal method is given as follows:

$$y_{j+1} = y_j + \frac{h}{2}\left(f(t_j, y_j) + f(t_{j+1}, y_{j+1})\right).$$

Find the region of absolute stability for the method.

Part 4. Answers

Derive the error bound for the composite midpoint rule

The Taylor expansion of f at $\bar{x} = (a+b)/2$ gives

$$f(x) = f(\bar{x}) + (x - \bar{x})f'(\bar{x}) + \frac{(x-\bar{x})^2}{2} f''(\xi)$$

where $\xi \in [\bar{x}, x]$. Then,

$$\int_a^b f(x)\, dx = (b-a)f(\bar{x}) + \frac{1}{2}\int_a^b (x-\bar{x})^2 f''(\xi)\, dx$$

where $\xi \in [a, b]$. This leads to the following

$$\left| \int_a^b f(x)\, dx - (b-a)f(\bar{x}) \right| = \frac{1}{2}\left| \int_a^b (x-\bar{x})^2 f''(\xi)\, dx \right|.$$

Derive the error bound for the composite midpoint rule

The RHS is given by

$$\begin{aligned}\frac{1}{2}\left|\int_a^b (x-\bar{x})^2 f''(\xi)\ dx\right| &\leq \frac{M}{2}\int_a^b (x-\bar{x})^2\ dx \\ &= \frac{M}{24}(b-a)^3\end{aligned}$$

where $M = \max_{x\in[a,b]}|f''(x)|$. Then, partition $[a,b]$ into N equidistant subintervals $[x_{j-1}, x_j]$ of length $h = x_j - x_{j-1} = (b-a)/N$ for $j = 1,\ldots,N$. $x_0 = a$, $x_N = b$, and $\bar{x}_j = x_{j-1} + \frac{1}{2}h$ for $j = 1,\ldots,N$.

$$\int_a^b f(x)\ dx = \sum_{j=1}^{N}\int_{x_{j-1}}^{x_j} f(x)\ dx \approx \sum_{j=1}^{N} hf\left(\bar{x}_j\right).$$

Derive the error bound for the composite midpoint rule

For each interval $[x_{j-1}, x_j]$,

$$\left| \int_{x_{j-1}}^{x_j} f(x)\, dx - hf\left(\bar{x}_j\right) \right| \leq \frac{M_j}{24} h^3$$

where $M_j = \max_{x \in [x_{j-1}, x_j]} |f''(x)|$. Let $M = \max_{1 \leq j \leq N} \{M_j\}$.

$$\left| \int_a^b f(x)\, dx - Q_{\text{mid}}(f) \right| \leq M \sum_{j=1}^{N} \frac{h^3}{24}.$$

Recall that $h = (b-a)/N$ and $hN = b - a$.

$$\left| \int_a^b f(x)\, dx - Q_{\text{mid}}(f) \right| \leq M \frac{(b-a)}{24} h^2.$$

Computational cost of matrix-vector multiplication

operation	# of $\times(/)$	# of $+(-)$	computational cost
$(\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^n) \mapsto \mathbf{x}^T\mathbf{y}$	n	$n-1$	$\mathcal{O}(n)$
$(\mathbf{x} \in \mathbb{R}^m, \mathbf{y} \in \mathbb{R}^n) \mapsto \mathbf{x}\mathbf{y}^T$	mn	0	$\mathcal{O}(mn)$
$(A \in \mathbb{R}^{m,k}, B \in \mathbb{R}^{k,n}) \mapsto AB$	mnk	$mn(k-1)$	$\mathcal{O}(mnk)$

Machine epsilon

Suppose that $x > 0$. We can write the real number x in the form

$$x = \mu \times b^{(q-p)},$$

where $b^{(p-1)} \leq \mu < b^p$. Since x lies between the floating point numbers $x_- = \mu_- \times b^{(q-p)}$ and $x_+ = \mu_+ \times b^{(q-p)}$, $\mathrm{fl}(x) = x_- \text{ or } x_+$.

$$|\mathrm{fl}(x) - x| \leq \frac{(x_+ - x_-)}{2} \leq \frac{1}{2} b^{(q-p)}.$$

Then,

$$\left| \frac{\mathrm{fl}(x) - x}{x} \right| \leq \frac{\frac{1}{2} b^{(q-p)}}{\mu \times b^{(q-p)}} \leq \frac{1}{2} b^{(1-p)}.$$

Roundoff error in the difference formula

The evaluation of $D(f,h)$ is given by

$$\hat{D}(f,h) = \frac{f(x_0+h)(1+\delta_{x_0+h})}{h} - \frac{f(x_0)(1+\delta_{x_0})}{h}.$$

Since

$$|\delta| < \frac{1}{2}\epsilon_{\mathrm{M}},$$

$$|D(f,h) - \hat{D}(f,h)| \le C\frac{\epsilon_{\mathrm{M}}}{h}$$

for some constant C.

Local truncation error

Compute the local truncation errors for the following methods:

- Implicit Euler method:

$$e_{j+1} = -\frac{h_j^2}{2} y''(\xi_j)$$

  where $\xi_j \in [t_j, t_{j+1}]$.

- Trapezoidal method

$$e_{j+1} = -\frac{h_j^3}{12} y'''(\xi_j)$$

  where $\xi_j \in [t_j, t_{j+1}]$.

Trapezoidal method for the stiff ODEs

$$y_{j+1} = y_j + \frac{h}{2}\left(-\lambda y_j - \lambda y_{j+1}\right).$$

Then,

$$y_{j+1} = y_j \left(\frac{1 - \frac{\lambda h}{2}}{1 + \frac{\lambda h}{2}}\right).$$

The numerical solution of trapezoidal method decreases to 0 for any step size $h > 0$ if $\lambda < 0$.